Kuhn-Munkres Parallel Genetic Algorithm for the Set Cover Problem and Its Application to Large-Scale Wireless Sensor Networks

Xin-Yuan Zhang, Yue-Jiao Gong, *Member*, *IEEE*, Zhi-Hui Zhan, *Member*, *IEEE*, Wei-Neng Chen, *Member*, *IEEE*, Yun Li, *Member*, *IEEE*, and Jun Zhang, *Senior Member*, *IEEE*

***Abstract*—Operating mode scheduling is crucial for the lifetime of wireless sensor networks. However, the growing scale of networks has made such a scheduling problem more and more challenging, as existing set cover and evolutionary algorithms become unable to provide satisfactory efficiency due to the curse of dimensionality. In this paper, a Kuhn-Munkres parallel genetic algorithm is developed to solve the set cover problem and is applied to lifetime maximization of large-scale wireless sensor networks. The proposed algorithm schedules the sensors into a number of disjoint complete cover sets and activates them in batch for energy conservation. It uses a divide-and-conquer strategy of dimensionality reduction, and the polynomial Kuhn-Munkres algorithm are hence adopted to splice the feasible solutions obtained in each subarea to enhance the search efficiency substantially. To further improve global efficiency, a redundant-trend sensor schedule strategy is developed. Additionally, we meliorate the evaluation function through penalizing incomplete cover sets, which speeds up convergence. Eight types of experiments are conducted on a distributed platform to test and inform the effectiveness of the proposed algorithm. The results show that it offers promising performance in terms of the convergence rate, solution quality, and success rate.**

***Index Terms*—parallel genetic algorithm, set cover problem, large-scale wireless sensor networks, Kuhn-Munkres algorithm.**

I. Introduction

WIRELESS sensor networks (WSNs) have been widely used in a number of fields to satisfy various requirements, such as road traffic monitoring [1], environmental observation [2], healthcare sensing [3], and asset monitoring [4]. Typically, hundreds or even thousands of sensors, each with a series of transceivers, a battery and a micro central processing unit, are deployed in a target area. Since it is impossible to recharge or replace the battery in some scenarios, how to extend the lifetime of WSNs becomes a critical task [5].

Manuscript received ...; revised …; accepted … This work was partially supported by the National Natural Science Foundation of China (NSFC) Key Project No. 61332002 and by the NSFC Youth Projects No. 61502542 and No. 61300044 (Y.-J. Gong and J. Zhang are equally contributed corresponding authors, email: gongyuejiao@gmail.com; junzhang@ieee.org).

X.-Y. Zhang, Y.-J. Gong, Z.-H. Zhan, W.-N. Chen, and J. Zhang are with the Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, China and with the Engineering Research Center of Supercomputing Engineering Software, Ministry of Education, China.

Y.-J. Gong is also with Department of Computer and Information Science, University of Macau, Macau.

Y. Li is with the School of Engineering, University of Glasgow, Glasgow G12 8QQ, U.K.

Existing ways for lifetime enhancement are classified into five categories: operation mode control [6], data processing [7][8], sink relocation [9]-[11], topology control [12][13], and optimal routing [14]-[16]. There are various definitions of the network lifetime. In this paper, the lifetime of a WSN refers to the duration of time that the network is able to carry out its set mission. Normally, the networks can fulfill its mission if it can guarantee the specified coverage requirements by the sensors deployed, i.e., the set cover condition is satisfied [17].

As summarized in [18]-[20], the deployment methods for sensors in WSNs vary with applications, which can be categorized into deterministic deployment and random deployment. Deterministic deployment is applied to a small- or medium-scale network in a friendly sensory environment [21]-[23]. The set cover problem here can be transformed into a minimum set cover problem or its dual problem. There are some certain theoretical developments [24]-[26] and optimization algorithms [18][21] related to this field. Since this problem is NP-hard, evolutionary-computation based solvers are potentially promising because of their powerfulness in dealing with NP-hard problems. However, the optimal number of sensors cannot be known in advance, which increases the difficulty of applying an evolutionary algorithm, such as the genetic algorithm (GA). The variable length chromosome puts a great challenge to the crossover operation of GA. Nevertheless, this issue has been well solved recently by using a bi-objective GA [27].

When the environment is inaccessible or unfriendly, or the number of sensors is too large, sensors are often scattered from an aircraft or by other means of transportation, which in effect results in random deployment. In order to guarantee coverage and connectivity, the sensors are to be densely deployed in target areas. To construct an energy-efficient WSN in this case, sensors are assigned to different cover sets independent of one another [28]. Activating them in batch ensures that only one cover set is active at a time and the others are scheduled to sleep. This scientific problem is known as the *Set K-cover* problem [29] or *Disjoint Set Covers* problem [30], which is a nondeterministic polynomial complete (NP) problem and hence its optimization is NP-hard.

A common objective of solving the *Set K-cover* problem is to maximize the lifetime of the WSN, but they differ slightly in terms of coverage constraints. In [31], sensors are aimed to be scheduled into K disjoint sets while guaranteeing that the coverage ratio of each set is as high as possible by modeling

the *Set K-cover* problem as an *N*-person card game, and solutions are obtained after a gaming process. A heuristic method, termed the Most Constrained-Minimally Constraining Covering (MCMCC), is proposed in [29]. The essence of MCMCC is to minimize the coverage of sparsely covered areas within one cover set. It requires that each cover set is able to cover the target area completely. We focus on complete cover sets in this paper.

Owing to their success in solving nondeterministic polynomial problems, GAs [32]-[34] and other evolutionary algorithms (EAs) [35]-[37] have been applied to the lifetime problem in WSNs recently. Lai *et al*. [38] propose a GA for maximum disjoint set covers (GAMDSC), which applies a scattering operator to the EA offspring to keep critical sensors from joining the same cover set. Hu *et al*. [39] propose a schedule transition hybridized genetic algorithm (STHGA), which adopts a forward encoding scheme for chromosomes and utilizes redundancy information via designing a series of transition operations. Ant-colony optimization for maximizing the number of connected cover (ACO-MNCC) is proposed in [40], which maximizes the lifetime of heterogeneous WSNs. These algorithms are shown competitive in solving small to medium sized WSNs. However, the performance of existing methods decreases substantially when dealing with a large-scale *Set K-cover Set* problem.

To improve set cover efficiency and large-scale WSN performance over existing algorithms, a Kuhn-Munkres scheduled parallel GA (KMSPGA) is developed in this paper, so as to provide the following features and benefits:

- A divide-and-conquer strategy to achieve dimensionality reduction in a simple but effective way and solve the small-scale *Set K-cover problem* separately in each subarea;
- The merging of local feasible solution is modeled as a Maximum Weight Perfect Matching (MWPM) problem such that Kuhn-Munkres (KM) algorithm [41][42] is applicable;
- A redundant-trend sensor schedule strategy (RTSS) to further improve global search efficiency, which is easier to implement than the existing auxiliary schedule strategies [39], and
- A modified fitness index by introducing a coverage inadequate penalty to guide convergence better.

The rest of this paper is organized as follows. In Section II, we describe the *Set K-cover* problem for WSNs, with assumptions and definitions given. In Section III, we develop KMSPGA in detail. This parallel genetic algorithm is comprehensively tested for performance through simulations in Section IV. Finally, conclusions are drawn in Section V.

II. Problem Formulation

A. Sensor Model

In this section, we introduce the sensor model adopted in our algorithm, which is crucial for the coverage, connectivity, and energy consumption.

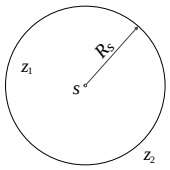


Fig. 1. Boolean Disk model. Considering two points z_1 and z_2 in the target area, we have $f(D(s, z_1)) = 1$ and $f(D(s, z_2)) = 0$.

A sensor coverage model is an abstract concept to measure the sensing capability and to quantify how well a sensor can monitor the occurrence of events within its sensing range. Various models have been proposed, such as the Boolean sector coverage model and the attenuated disk coverage model [43]. Comparisons of existing models are made in [44]-[46].

In this paper, we adopt the Boolean disk model because of its concise definition and wide applicability [19][43], which is illustrated in Fig. 1. This model ignores the dependency of environmental conditions. A point in the space of the target area is considered to be covered only if it is within the range of at least one sensor. A coverage function is formalized as:

$$f(D(s,z)) = \begin{cases} 1, & \text{if } D(s,z) \le R_s \\ 0, & otherwise \end{cases} \quad (1)$$

$$D(s,z) = \sqrt{(s_x - z_x)^2 + (s_y - z_y)^2} \quad (2)$$

where $s = (s_x, s_y)$ is the central coordinate of a sensor and (z_x, z_y) is the coordinate of point z in a 2-dimensional space, R_s is the sensing radius and $D(s, z)$ calculates the Euclidean distance between s and z. Once a point is covered by a sensor, the coverage function value is 1, otherwise, it is 0. In [47], a geometric analysis of the relationship between coverage and connectivity is provided, showing that the connectivity can be guaranteed inherently if 1) coverage is satisfied and 2) the communication range of sensors is no less than twice of the sensing range. Afterwards, many related publications for solving the *Set K-cover* problems are based on this proof [31][39] that they use the sensors with communication range at least twice than the sensing range and consider the coverage constraint only. In this paper, we also make such an assumption and meet the connectivity constraint by satisfying the coverage constraint.

In WSNs, the energy consumption of a specific sensor can be determined by the following aspects: 1) the communication when sending and receiving data; 2) the data type and size acquired for processing; 3) the differences in the inherent characteristic of the battery; 4) the different local working condition, such as temperature. These are, however, diverse and highly related to the application scene. In this paper, we focus on the *Set K-cover* problem. Instead of fully formulating the above issues influencing the energy consumption, we simplify the model by making the following three assumptions: 1) for every active sensor, the amount of energy consumption is identical per unit time; 2) for every sleep sensor, they consume negligible energy compared with the active ones; 3) every sensor carries the same amount of energy in the initial stage. Based on these assumptions, the lifetime of each complete cover set is identical to each one another. Nevertheless, it is to be noted that the energy consumption model in use have no direct relationship with the operations or parameters of our proposed KMSPGA. In practical application,

there is still room for KMSPGA to adapt this framework to fitting new forms of energy constraints.

B. Set K-Cover Problem

In this paper, we focus on maximizing the lifetime of a large-scale WSN using a static optimization strategy. The proposed KMSPGA is an off-line algorithm that pre-calculates K disjoint complete cover sets at a time. By alternatively activating the K cover sets in batch, the lifetime of the WSN can be K times larger than the lifetime of a single cover set. Hence, maximizing the lifetime of the WSN is equivalent to finding the maximum disjoint complete cove sets.

We consider the *Set K-cover* problem a scheduling problem. For a positive integer K, sensors are judiciously scheduled into K disjoint cover sets such that each cover set is able to meet the coverage requirement. In this paper, we focus on a complete coverage. The premise of finding K disjoint complete cover sets is that every element of the target area is covered by at least K sensors. Sufficiency of this premise is proven in the following of this subsection. Fig. 2 illustrates the concept of an element and a K-covered element. In Fig. 2 (a), the grey square area is divided into eight elements. Element E_x in Fig. 2 (b) is covered by four sensors; hence it is called 4-covered.

Without loss of generality, assume that a target area Γ is a rectangle and that N sensors $S_1, S_2, \ldots, S_N$ are randomly deployed in Γ. A constraint of the coverage requirements is a complete coverage. Given a positive number K, sensors are scheduled into K cover sets $C = \{C_1, C_2, \ldots, C_K\}$. For each cover set C_i ($i = \{1, 2, \ldots, K\}$), if every element of Γ is covered by at least one sensor in C_i, then C would be considered as a feasible solution of the *Set K-cover* problem. The *Set K-cover* problem can be formalized as:

$$\bigcup\nolimits_{\forall S_k \in C_i} E(S_k) \supseteq \Gamma \tag{3}$$

$$\bigcup_{i=1}^{K} C_i \subseteq S \tag{4}$$

$$C_i \bigcap C_j = \varnothing,\ i \neq j,\ i, j \in \{1, 2, ..., K\} \tag{5}$$

where $E(S_k)$ represents the element sensed by sensor S_k, k is the sensor index, S is the collection of sensors. Assume that the target area is partitioned into M elements $E_1, E_2, \ldots, E_M$. To make a clear explanation of how to calculate the upper bound of K, we first give the following proposition and its proof:

Proposition 1: The prerequisite of finding K disjoint cover sets is that each element is covered by at least K sensors.

Proof: Assume that E_τ is covered by Q sensors $SC_\tau = \{S_{\tau,1}, S_{\tau,2}, \ldots, S_{\tau,Q}\}$, where SC_τ is the collection of sensors covering element E_τ. There still exist K disjoint complete cover sets with $Q < K$.

Let C be a feasible solution of the *Set K-cover* problem. Then, we have $|C| = K$. Considering the same element E_τ in the assumption, E_τ is expected to be covered in each $C_i \in C$ according to (3). Then, K cover sets are considered as K covering tasks, and we have Q sensors that can perform this task. Then, K tasks are assigned to Q sensors. Since we have $|C| = K > |SC_\tau| = Q$, there exist cover sets C_p, $C_q \in C$, and sensor $S_{\tau,m} \in SC_\tau$ ($m \in \{1, 2, \ldots, Q\}$) satisfying $S_{\tau,m} \in C_p \cap C_q$ according to the drawer principle, and therefore it contradicts

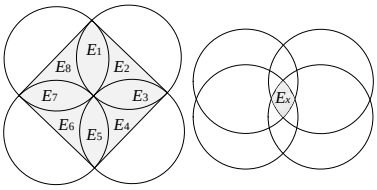


Fig. 2. Notion of an element and a K-covered element. (a) The shaped area is divided into eight elements $E_1, E_2, \ldots, E_8$, where four elements are covered by two sensors and the remaining four by one only. (b) Element E_x is called K-covered, if it is covered by K sensors, in which case $K = 4$.

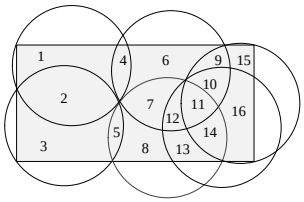


Fig. 3. Elements E_1, E_3, E_6, E_8, and E_{15} are called critical elements, as they are covered by the minimum number of sensors.

(5). In conclusion, the assumption is invalid and hence the proposition is proven to be tenable.

In order to better explain how to estimate the upper limit of K, the notion of critical element and critical sensor is introduced as follows. The target area is partitioned into a number of elements by thousands of densely deployed sensors. An element covered by the minimum number of sensors is called a 'critical element' and the corresponding sensors 'critical sensors'. Fig. 3 illustrates critical elements and critical sensors, where six sensors divide the rectangular area into sixteen elements E_1-E_{16}. Being covered by one sensor only, elements E_1, E_3, E_6, E_8, and E_{15} are critical elements.

Let E_c be a critical element. Assume that the number of sensors covering E_c is $\hat{U}$. According to *Proposition* 1, we can at most find $\hat{U}$ disjoint complete cover sets only if the $\hat{U}$ critical sensors are chosen in $\hat{U}$ disjoint cover sets which guarantee that E_c is covered by every cover set, and therefore $\hat{U}$ is regarded as the upper limit of K.

C. Critical Parameters

In this subsection, we discuss some critical parameters related to a large-scale WSN. A redundant rate represents the density of sensors deployed in the target area. The redundant rate in the 2D ideal plane model is computed as (6) according to [39], where $area(\Gamma)$ is the area of Γ and N is the number of sensors.

$$\eta = \frac{N \cdot \pi \cdot R_s^{\,2}}{\hat{U} \cdot area(\Gamma)} \tag{6}$$

It is difficult to compute the coverage ratio of Γ accurately when applying the Boolean disk model. For this reason, we divide the target area into T smaller square grids ($g_1, g_2, \ldots, g_T$), T being computed as (7), where d is the width of the grid. A coverage ratio is defined as (8), where $N_g(S_i)$ represents the collection of grids covered by sensor S_i and $|N_g(S_i)|$ is the number of grids in $N_g(S_i)$. Equation (9) indicates that a grid belongs to no more than one collection in case of a repeat count.

$$T = \frac{area(\Gamma)}{d^2} \tag{7}$$

$$\delta = \frac{1}{T}\sum_{i=1}^{N} |\, N_g(S_i)\,| \tag{8}$$

$$N_g(S_i) \bigcap N_g(S_j) = \varnothing, \quad i \neq j,\ i, j \in \{1,2,...,N\} \tag{9}$$

The coverage criteria stipulates that grid g_j is covered by sensor S_i only if all its four vertices are within the sensing range of S_i. Fig. 4 shows how to calculate the coverage ratio. This calculation method is widely used to estimate the coverage ratio [39][40]. However, the grid width can influence the computation of the coverage ratio in terms of computational complexity and accuracy. Fig. 5 gives an example to explain this special case, where the grey grid is apparently covered by the WSN. Unfortunately, it is regarded as uncovered due to the above coverage criteria whether a grid is covered. If the width of this grid is halved, two resultant grids become covered. However, the calculation of the coverage ratio is of an $O(N \times T)$ computational complexity. A shorter width means a higher computational complexity according to (7).

In our work, we adopt a simple strategy to determine the grid width in a preprocessing step. Given Ψ kinds of d_i (i = {1, 2, …, Ψ}, $d_i < d_{i+1}$) in the process of estimating the upper limit of K, we choose the smallest d_1 first to obtain an exact value $\hat{U}_1$, because d_1 is small enough to guarantee accuracy. Then d_i (i = {2, 3, …, Ψ}) is adopted to work out $\hat{U}_i$ in sequence. The largest d_i (ensuring $\hat{U}_i = \hat{U}_1$) is used for calculating the coverage ratio. **Algorithm 1** presents a set of pseudocode of this preprocess of choosing a proper grid width. In this paper, we adopt 5 kinds of grid widths: (d_1, d_2, d_3, d_4, d_5) = (0.625, 0.78125, 1, 1.25, 1.5625).

III. Proposed Parallel Genetic Algorithm

A. Kuhn-Munkres Parallel Genetic Approach

KMSPGA is designed on a divide-and-conquer strategy, and the polynomial KM algorithm is adopted to splice the feasible solutions obtained in each subarea. The framework of KMSPGA is shown in Fig. 6. In the first step, we uniformly divide the target area into a number of subareas and encode them. Assuming that the number of sensors within subarea A_i is N_i, it satisfies:

$$\sum_{i=1}^{L\times W} N_i = N \tag{10}$$

where L and W denote the number of partitions along horizontal and vertical directions respectively, and hence $L \times W$ denotes the number of subareas obtained. Therefore, the *Set K-cover* problem size of subarea A_i is whittled down to N_i. After the partition process, the small-scale *Set K-cover* problem within each subarea is separately solved by a parallel processing module. When local solutions of each small-scale *Set K-cover* problem reach a predefined state, they are spliced through a KM combination operation to achieve global optimization efficiently. There are two termination conditions: 1) FEs reaches its upper limit and/or 2) the number of complete cover sets reach $\hat{U}$. KMSPGA deploys a *termination-controller* in its master process. The controller checks whether the current process has reached the termination condition at the end of every generation. If so, the master process broadcasts a termination signal to all the slave

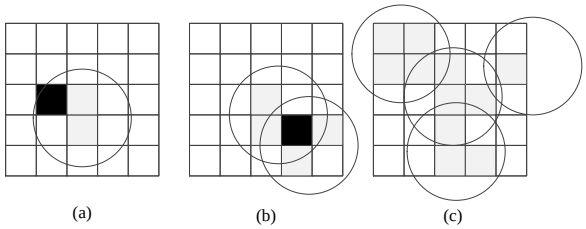


Fig. 4. Calculation of the coverage ratio. (a) The light grey grids are covered while the black one is not covered because one of its vertices is beyond the sensing range of the sensor. (b) The black grid is covered by two sensors S_p and S_q, but only belongs to either of $N_g(S_i)$, $i \in \{p, q\}$, in the case of a repeat count. (c) The coverage ratio is 11/25 = 0.44.

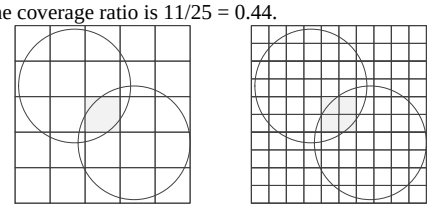


Fig. 5. An example to show how the grid width influences the computation of the coverage ratio. (a) The grey grid is regarded as uncovered due to the coverage criteria. (b) The grey grid becomes covered by either of the two sensors after the grid width is shortened.

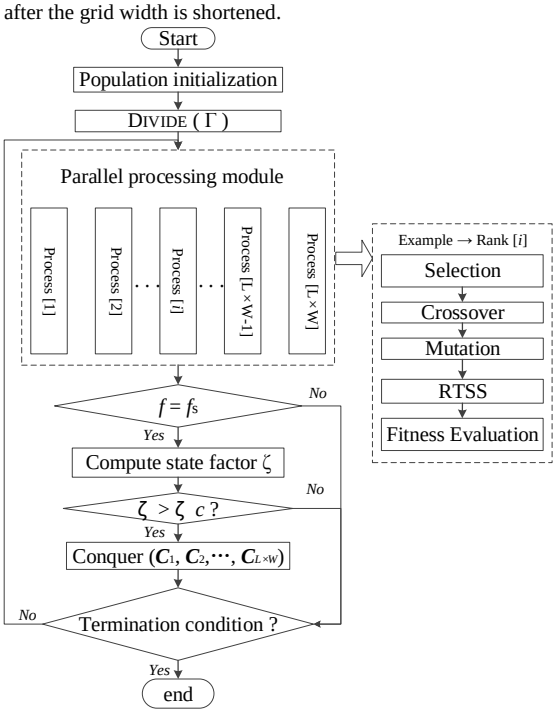


Fig. 6. Flowchart of the Kuhn-Munkres parallel GA framework.

Algorithm 1 Preprocessing of choosing a proper grid width

```
Procedure WIDTHCHOOSE {d_1,…,d_Ψ}
    d ← d_1 ;
    Compute Û_1 ;
    for i = Ψ → 2 do
        d ← d_i ;
        Compute Û_i
        if Û_i = Û_1 then
            d ← d_i ;
            break ;
        end if
    end for
    return d ;
end procedure
```

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2511142, IEEE Transactions on Evolutionary Computation

processes and, afterwards, all the processes of KMSPGA terminate. Pseudocode of this divide-and-conquer strategy is given in **Algorithm 2**.

Divide() contains two steps. Firstly, the target area Γ is uniformly partitioned into $L \times W$ subareas $\boldsymbol{A} = (A_1, A_2, \ldots, A_{L\times W})$. Fig. 7 gives four examples where Γ is divided into 2, 8, 32, and $L \times W$ subareas in Fig. 7 (a), (b), (c), and (d), respectively. Then, the centers of all sensors are traversed to obtain a classification $\boldsymbol{SE} = \{SE_1, SE_2, \ldots, SE_{L\times W}\}$, where SE_i is the collection of sensors falling in subarea A_i. Every sensor within SE_i satisfies that its central coordinate $(s_x, s_y) \in A_i$ ($k = \{1, 2, \ldots, N_i\}$). Additionally, if the center of a sensor falls on the boundaries of two or more subareas, it will be randomly scheduled into any one of the subareas. In order to keep the concision of KMSPGA, we adopt a uniform partition here instead of other ways such as clustering techniques. Besides, the uniform partition is convenient for the following combination operation.

In the parallel processing module, each process evolves a sub-population to obtain a feasible solution. The collection of sub-populations is formulized as $\boldsymbol{SP} = (SP_1, SP_2, \ldots, SP_{L\times W})$. Each sub-population size is N_p. They are evolved independently through a selection, crossover, mutation, and RTSS operation in *processing*(). We estimate the state of each sub-population through periodically sampling the information of the best individual at a sampling frequency f_s. A state factor ζ is computed as follows:

$$\zeta = \frac{1}{\hat{\mathrm{U}}} \sum_{i=1}^{L\times W} U_i \tag{11}$$

where U_i is the number of disjoint complete cover sets obtained by the best individual of SP_i. Since $U_i \le \hat{\mathrm{U}}$ ($i = 0, 1, \ldots, L\times W$), we have $\zeta \le L\times W$. Therefore, the upper limit of the threshold value, ζ_c, is set to $L\times W$–1. The value of ζ determines whether the KM combination operation will be applied. The independent evolution process will be terminated until ζ reaches ζ_c. Therefore, instead of performing the KM operation every generation, the execution timing of KM is adaptively adjusted based on the state factor. This way, the effectiveness of the operation is improved, and hence the computational cost is substantially reduced. The threshold ζ_c determines the frequency of performing the KM operation (merging the local feasible solutions) and then checking for the termination condition. This procedure however does not affect the main evolution process of solutions to much extent. Thus, different settings of ζ_c will not change the output solution quality of the proposed algorithm, but only influence the execution time. The standard uniform crossover and uniform mutation are adopted in the evolutionary process. RTSS is conducted right after mutation, and before fitness evaluation, which is introduced in detail in Subsection *D*. The tournament selection is adopted because of its efficiency, with a tournament size T_s.

Conquer() is a combination operation in order to merge the feasible local solutions $\boldsymbol{C} = (\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_{L\times W})^{\mathrm{T}}$, $\boldsymbol{C}_i = (C_{i,1}, C_{i,2}, \ldots, C_{i,\hat{\mathrm{U}}})$. $\boldsymbol{C}_i$ is the best solution of the small-scale *Set K-cover* in subarea A_i. Fig. 8 shows an example of this merging process, where Γ is divided into 2×4 subareas. We need three steps to merge $(\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_8)$ into $\boldsymbol{C}_{1\text{-}8}$. Considering a couple of neighboring sub-populations *p* and *q*, *Conquer*() is expected

Algorithm 2 Process of divide-and-conquer.

```
1: Procedure DIVIDE (Γ)
2:   Divide Γ into L×W subareas (A_1, A_2, …, A_{L×W})
3:     for i = 1 → N do
4:       for j = 1 → L×W
5:         if S_i falling in A_j then
6:           Add S_i into SE_j;
7:         end if
8:       end for
9:     end for
10: end procedure
1: Procedure PARALLELPROCESSING (SP)
2:   DIVIDE (Γ) ;
3:   Repeat
4:     for process = 1 → L×W parallel do
5:       processing (P_process); // g_c is the current generation
6:       if g_c % f_s = 0 then  // "%" is a modulus operator
7:         compute the state factor ζ.
8:         if ζ > ζ_c then
9:           CONQUER (C_1, C_2, …, C_{L×W}) ;
10:        end if
11       end if
12:    end for
13:  Until termination conditions are satisfied.
14: end procedure
1: Procedure CONQUER (C_1 , C_2, …, C_{L×W}).
2:   Repeat
3:   Compute the weight matrix w_E.
4:   Combine neighbor cover sets (C_p, C_q) ;
5:   Until all local cover sets are merged into a global one.
6: end procedure
```

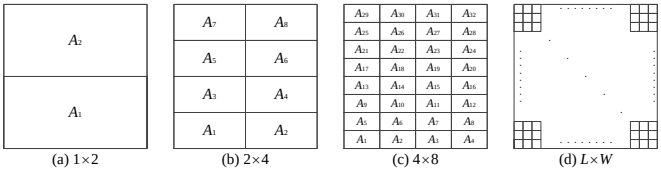


Fig. 7. Division of the target area and encoding of the subareas. The target area is divided into 2, 8, 32, and $L \times W$ subareas in figure (a), (b), (c), and (d).

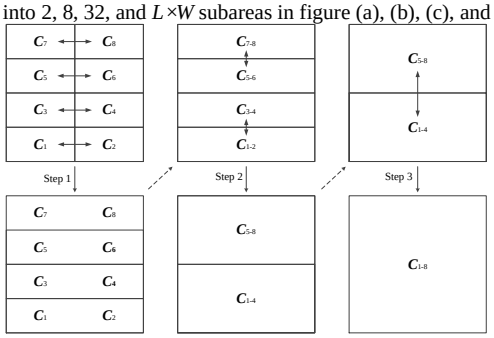


Fig. 8. Combination of the subareas.

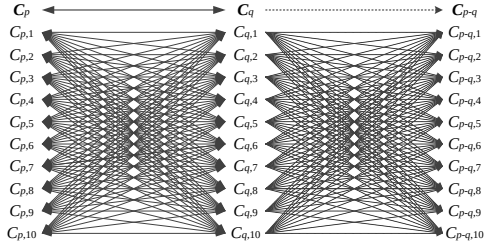


Fig. 9. Bipartite combinations of solutions $\boldsymbol{C}_p$ and $\boldsymbol{C}_q$ obtained by neighbor sub-populations, such that the Kuhn-Munkres algorithm can be applied.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2511142, IEEE Transactions on Evolutionary Computation

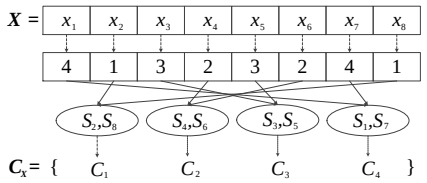


Fig. 10. Representation of chromosome. $\boldsymbol{C}$ is an equivalent way of representing a chromosome, where genes of the same value form a cover set.

to find a best combination $\boldsymbol{C}_{p\text{-}q}$ for each dimension of $\boldsymbol{C}_p$ and $\boldsymbol{C}_q$ ensuring that the coverage ratio summation of $C_{p\text{-}q,k}$ is maximal. The total number of matching combinations is Û!. As shown in Fig. 9, given Û = 10, we have 10! kinds of matching ways of $\boldsymbol{C}_p$ and $\boldsymbol{C}_q$. This combination problem can be modeled as a MWPM problem in graph theory, which can now be solved using the KM algorithm.

In the literature, KM algorithm has been successfully applied to a number of fields, such as allocation of vehicle-to-infrastructure and vehicle-to- vehicle links [48], group role assignment [49], and user grouping for grouped OFDM-IDMA [50].Given a bipartite graph $G = (\boldsymbol{V}, \boldsymbol{E})$ and weight function $w(e)$, MWPM aims at finding a perfect matching of maximum weight. The weight of the matching M is formulized as (12). Cover sets ($C_{i,1}$, $C_{i,2}$, …, $C_{i,\hat{U}}$) are considered as the vertices of G ($i \in \{p, q\}$). Weight $w_{i,j}$ of edge e between $C_{p,i}$ and $C_{q,j}$ is computed as (13), where $|N_g(C_{p,i} \cup C_{q,j})|$ represents the number of grids covered by the sensors within $C_{p,i}$ and $C_{q,j}$, and $|N_g(A_p \cup A_q)|$ is the number of grids covered within A_p and A_q. Then, the weight matrix $\boldsymbol{W}_{\boldsymbol{E}}$ is represented as (14). KM combination is constantly conducted until all local solutions are totally combined into a global solution.

$$w(M) = \sum_{e_k \in M} w(e_k) \tag{12}$$

$$w_{i,j} = \frac{|N_g(C_{p,i} \cup C_{q,j})|}{|N_g(A_p \cup A_q)|} \tag{13}$$

$$\boldsymbol{W}_{\boldsymbol{E}} = \begin{pmatrix} w_{1,1} & \cdots & w_{1,\hat{U}} \\ \vdots & \ddots & \vdots \\ w_{\hat{U},1} & \cdots & w_{\hat{U},\hat{U}} \end{pmatrix} \tag{14}$$

B. Chromosome Representation

In this subsection, we describe the chromosome representation in KMSPGA. We compute the value of Û in the whole target area. It is worth mentioning that even if we divided the target area into subareas, we still have to ensure that the whole target area can be covered by Û complete cover sets. Therefore, Û is also the upper limit of the number of complete cover sets for each subarea, then, sub populations SP_1, SP_2, …, $SP_{L\times W}$ share the same Û. Taking Population SP_k as an example, each chromosome is encoded as $\boldsymbol{X} = (x_1, x_2, \ldots, x_n)$, where x_i represents the batch number of sensor i, and n is the number of sensors falling in A_k ($n = N_k$). Since there is at most Û batches, we have $x_i \in \{1, 2, \ldots, \hat{U}\}$.

For chromosome $\boldsymbol{X}$, sensors with same batch number are chosen to a same cover set. Therefore, $\boldsymbol{X}$ is transformed into $\boldsymbol{C}_{\boldsymbol{X}} = (C_1, C_2, \ldots, C_{\hat{U}})$, which is a candidate solution of *Set K-cover* problem. Fig. 10 gives an example of chromosome representation and shows the relationship between $\boldsymbol{X}$ and $\boldsymbol{C}_{\boldsymbol{X}}$, where n is eight and Û is four. On the contrary, $\boldsymbol{X}$ can be easily transformed from $\boldsymbol{C}_{\boldsymbol{X}}$. Therefore, $\boldsymbol{X}$ and $\boldsymbol{C}_{\boldsymbol{X}}$ are equivalent on representing an individual. In the remainder of this section, we adopt the C_X structure in representing a chromosome because this form is more convenient for introducing and descripting the operations of KMSPGA while X is adopted in the practical implementation of KMSPGA.

Algorithm 3 Fitness evaluation
SF_k: a flag variable in case of repeat count.
RF_i: a flag variable representing whether S_i is redundant.

```
1:  Procedure CHROMOSOMEEVALUATION (x_1,…,x_N)
2:      for i = 1 → N do
3:          SF_i ← 0;
4:          RF_i ← true; // S_i is redundant if RF_i is true
5:      end for
6:      for i = 1 → T do
7:          for j = 1 → N do
8:              k ← x_j ;
9:              if g_i ⊂ N_g(S_j) and SF_k = 0 then
10:                 CN_k ← CN_k+1 ;
11:                 SF_k ← 1;
12:                 RF_j ← false;
13:             end if ;
14:         end for
15:     end for
16:     f ← 0;
17:     for i = 1 → Û do
18:         δ_i ←CN_i·T^-1 ;
19:         f ← f +δ_i·P(δ_i) ;
20:     end for
21: end procedure
```

C. Improved Fitness Index

In STHGA [39], the evaluation function is defined as (15), where δ_i represents the coverage ratio of cover set C_i. The computation of δ_i is shown in (16), where the value of $\delta_{i,k}$ is 1 if grid k is covered by C_i. The value of δ_i ($i \in \{1, 2, \ldots, \hat{U}\text{-}1\}$) is 1 in STHGA because of the forward encoding scheme. The evaluation function of GAMDSC [38] is shown in (18), where f_B represents the number of disjoint complete cover sets and $\lfloor x \rfloor$ denotes the floor of x.

$$f_A = \frac{1}{T}\sum_{i=1}^{\hat{U}} \delta_i \tag{15}$$

$$\delta_i = \frac{L \cdot W}{T} \sum_{k=1}^{T\cdot(L\cdot W)^{-1}} \delta_{i,k} \tag{16}$$

$$\delta_{i,j} = \begin{cases} 1, & \textit{if grid } j \textit{ is covered by } C_i \\ 0, & \textit{otherwise} \end{cases} \tag{17}$$

$$f_B = \sum_{i=1}^{\hat{U}} \lfloor \delta_i \rfloor \tag{18}$$

In order to improve the convergence rate, we adopt a penalty function $P(\delta_i)$ of (19), where λ is the penalty coefficient. The fitness evaluation function of KMSPGA is given in (20). Hence, the contribution of an incomplete cover set is lower than a complete one because of this penalty. Therefore, individuals with more incomplete cover sets are

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2511142, IEEE Transactions on Evolutionary Computation

eliminated more easily than those with more complete cover sets. **Algorithm 3** shows the pseudocode of the fitness evaluation in KMSPGA.

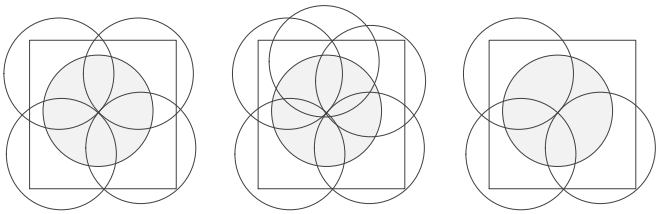
$$P(\delta_i) = \begin{cases} 1, & if\ d_i = 1 \\ \lambda, & otherwise \end{cases} \tag{19}$$

$$f = \sum_{i=1}^{\hat{U}} \delta_i \cdot P(\delta_i) \tag{20}$$

D. Redundant-Trend Sensors Schedule Operation

In RTSS, the redundant information is indirectly utilized in order to improve search efficiency. The redundant information is collected in the fitness evaluation process. In **Algorithm 3**, steps 2 to 15 give this collection process. As can be noted from the pseudocode, the collection process is embedded in the fitness evaluation in case of increasing computational complexity. Note that after collecting the redundant information, RTSS is not applied directly after the fitness evaluation, but, as introduced in subsection *A*, it is performed after crossover and mutation operations, the landscape of the chromosome may change. Thus, the redundant information utilized in RTSS is hysteretic.

Considering that S_k is a member of C_j, whether S_k is redundant for C_j depending on its contributions to C_j. S_k is considered to be redundant only if it has no contributions to C_j, which is judged in the fitness evaluation. However, S_k may not still be redundant, because crossover and mutation operations may change the members of C_j. Therefore, S_k is called redundant-trend sensors in RTSS due to this uncertainty of the redundant state. Fig. 11 shows an example of this uncertainty. The grey sensor is considered to be redundant after the fitness evaluation operation. However, it is uncertain whether it is still redundant after crossover or mutation.



(a) (b) (c)

Fig. 11. Illustration of a redundant-trend sensor, with a redundant state uncertain after crossover and mutation operations. (a) Since the area covered by this sensor has already been covered by other sensors in the same cover set, the grey sensor is considered to be a redundant one. Figure (b) and (c) show two situations of redundant state of the grey sensor after the crossover and mutation operations. The grey sensor is redundant in (b) while it is no longer redundant in (c).

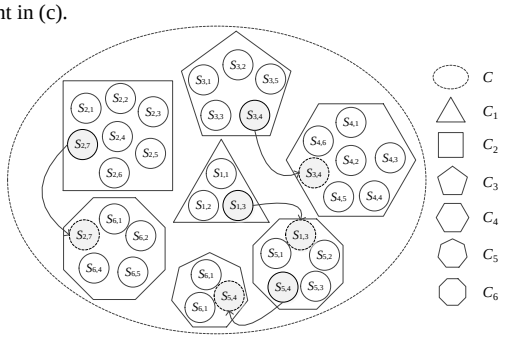


Fig. 12. Redundant-trend sensors transition between disjoint cover sets, where grey sensors are redundant-trend sensors. Different cover sets are represented

The process of RTSS is described as follows. Suppose that the cover set is $C = \{C_1, C_2, \ldots, C_{\hat{U}}\}$. Firstly, we traverse the redundant information of the sensors in C_i. Assuming $S_{i,k}$ is the kth member of C_i, if $S_{i,k}$ is judged to be redundant for C_i in **Algorithm 3**, we then consider it a redundant-trend sensor in RTSS. A cover set C_m ($m \in \{1, 2, \ldots, \hat{U}\}$) will receive $S_{i,k}$ through a tournament selection, where N_c candidates are randomly selected and the one with the lowest coverage ratio is chosen to receive $S_{i,k}$ as one of its members. Fig. 12 illustrates this schedule strategy between disjoint cover sets. In Fig. 12, different polygons represent different cover sets, the grey sensors are redundant-trend sensors, and the direction of arrow represents the schedule direction. RTSS has twofold functions. It helps enhance the coverage ratio through the schedule strategy if the redundant-trend sensor is actually a redundant one. However, if the redundant sensor is no longer redundant for the current cover set, the scheduling operation becomes a disturbance for the population. This kind of stochastic disturbance enriches the diversity of the population.

IV. Experimental Results

A. Experimental Setup

In this section, experiments are conducted to ascertain the performance of KMSPGA. In Subsection *B*, we compare KMSPGA with the state-of-the-art algorithms. MCMCC, GAMDSC, and STHGA are serial algorithms which perform well in solving the *Set K-cover* problem. The experiments and comparisons are used to verify the effectiveness of our proposed KMSPGA algorithm for lifetime maximization of large-scale WSNs. In Subsection *C*, we compare KMSPGA with a traditional pure parallel genetic algorithm (PGA). Further, the performance of the PGA embedding only RTSS (SPGA) or KM combination (KMPGA) are also tested in order to study the effectiveness of the two operations. Experiments in Subsection *D* and *E* are designed to evaluate the robustness of KMSPGA with different partitions and the redundant rates. In Subsection *F*, we conduct parameter investigation and give their suggested values. Finally, experiments in Subsections *G-I* are conducted with new and different testing scenarios to further verify the performance of KMSPGA.

KMSPGA and other algorithms are tested on a computer cluster of 25 nodes (with a total of 100 processing cores), which is homogenous with the same Intel core i3-3240 CPU running at 3.40 GHz, 4GB memory and Ubuntu 12.04 LTS

TABLE I
THE PARAMETER SETTING OF KMSPGA

Parameter	Description	Value
Ψ	Number of grid width classification.	5
λ	Penalty coefficient in cost evaluation.	0.2
L	Number of partitions along horizontal direction.	{2,4}
W	Number of partitions along vertical direction.	{2,4,8}
f_s	Sampling frequency.	3000
P_c	Crossover probability.	0.6
P_m	Mutation probability.	0.001
N_c	Number of candidates in RTSS.	$[3^{-1}\hat{U}]$
N_p	Sub-population size.	15
T_s	Tournament size	5
ζ_c	Threshold of the state factor with 2×2 partitions.	3.0
	Threshold of the state factor with 2×4 partitions.	7.0
	Threshold of the state factor with 4×4 partitions.	15.0
	Threshold of the state factor with 4×8 partitions.	30.0

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2511142, IEEE Transactions on Evolutionary Computation

TABLE II
EXPERIMENT ON 12 INSTANCES IN COMPARISON WITH SERIAL ALGORITHMS

INSTANCE	N	R_s	η	Û	GAMDSC					STHGA					KMSPGA			
					FEs	*U*		*t-test*[1]	S_r	FEs	*U*		*t-test*[2]	S_r	FEs	*U*		S_r
					Mean	*Mean*	*Std*	*p-value*		*Mean*	*Mean*	*Std*	*p-value*		*Mean*	*Mean*	*std*	
I1-1	5000	5	4.760	33	15000	2.0	0	0	0	10248	**33**	0	*N/A*	**1.00**	**2688**	**33**	0	**1.00**
I1-2		8	4.845	83	15000	7.4	0.67	3.31e-61	0	11507	82.9	0.25	1.6e-01	0.93	**2922**	**83**	0	**1.00**
I2-1	10000	5	4.488	70	30000	1.8	0.50	4.10e-65	0	25466	69.9	0.25	1.6e-01	0.93	**8418**	**70**	0	**1.00**
I2-2		8	4.493	179	30000	10.2	0.57	1.76e-73	0	27077	178.8	0.38	2e-02	0.83	**9324**	**179**	0	**1.00**
I3-1	15000	5	4.446	106	45000	1.3	0.50	1.15e-61	0	44887	103.9	1.13	8.90e-09	0.03	**28350**	**105.8**	0.55	**0.87**
I3-2		8	4.435	272	45000	12.5	0.57	6.76e-79	0	40946	271.8	0.50	2e-02	0.8	**15488**	**272**	0	**1.00**
I4-1	20000	5	4.303	146	60000	1.1	0.35	9.21e-76	0	59956	143.7	1.33	1.75e-09	0.06	**33086**	**145.9**	0.25	**0.93**
I4-2		8	4.347	370	60000	13.2	0.76	3.12e-79	0	59802	367.9	1.36	3.7e-00	0.1	**35838**	**370**	0	**1.00**
I5-1	25000	5	4.462	176	75000	1.6	0.51	9.25e-62	0	72679	175.4	0.63	3.8e-01	0.5	**26761**	**175.7**	1.27	**0.93**
I5-2		8	4.458	451	75000	16.1	0.57	2.46e-85	0	74496	449.7	0.87	8.22e-09	0.2	**36955**	**451**	0	**1.00**
I6-1	30000	5	4.446	212	90000	2.5	0.50	1.032e-77	0	87592	211.1	0.74	1.35e-07	0.3	**22845**	**212**	0	**1.00**
I6-2		8	4.570	528	90000	20.8	0.92	3.30e-81	0	86594	527.6	0.62	1e-02	0.67	**34785**	**528**	0	**1.00**

1 Two tailed t-test of the null hypothesis that GAMDSC is equal to KMSPGA. 2 Two tailed t-test of the null hypothesis that STHGA is equal to KMSPGA.

64-bit operating system. The parallel programming practice uses the Message Passing Interface (MPI). Table I shows the parameter settings. P_c and P_m, the constant crossover and mutation rates, are set as $P_c = 0.6$ and $P_m = 0.001$, respectively. The value of λ determines the degree of punishment for the individuals with incomplete cover sets. In all of the test instances, we adopt $\lambda = 0.2$ as default value. N_c, the number of candidates in RTSS, is empirically setting to 3^{-1}Û. Sample frequency f_s is set to 3000 function evaluations (FEs). In Subsections *B*-*H*, Γ is a 50×50 square area, and in Subsection *I*, Γ is a 3-D surface. $P_{L\times W}$ denotes the partition way of Γ. ζ_c is set to 3.0, 7.0, 15.0, and 30.0 for $P_{2\times2}$, $P_{2\times4}$, $P_{4\times4}$, and $P_{4\times8}$, respectively. Thirty trials are performed for each instance and the results are averaged over the trials. A two-tailed *t*-test of the null hypothesis is conducted in Subsections *B*, *G*, *H*, and *I*. The null hypothesis will be rejected if *p-value* is smaller than the significance level $\alpha = 0.05$.

B. Comparison with State-of-the-Art Serial Algorithms

We conduct experiments on 12 instances to verify the performance of KMSPGA in comparison with the serial algorithms: MCMCC, GAMDSC, and STHGA. $P_{2\times4}$ is adopted in this subsection.

The experimental results are listed in Table II. *Mean* and *Std* are the mean quality and standard deviation. S_r is the success rate. The best solutions are marked in bold. KMSPGA outperforms the other algorithms in terms of the convergence rate and solution quality. Furthermore, KMSPGA produces significant increases both in the convergence rate and in the success rate in the higher dimensional space. The success rate of instances I1-1, I1-2, I2-1, I2-2, I3-2, I4-2, I5-2, I6-1, and I6-2 reach 1.00. The mean FEs is far less than the serial algorithms. MCMCC is not available because it took an unacceptable time before termination. The worst runtime of MCMCC is $O(N^2)$ [39], where N is the number of sensors. This heuristic method performs efficiently when the number of sensors is small or medium. However, the computational complexity of MCMCC becomes so high in terms of large-scale WSNs that it fails to work out a feasible solution in an acceptable time, i.e., I2-1, a single run of MCMCC exceeds eight hours. GAMDSC is another genetic algorithm used in this experiment. Since GAMDSC lacks an efficient search strategy to handle such a large number of sensors, solutions obtained by GAMDSC are fewer than Û. STHGA possesses high quality of solutions in the low-dimensional spaces when the number of sensors is less than 5000. The complicated local search operations help STHGA search the problem space efficiently. However, the performance of this serial GA is badly influenced by curse of dimensionality. The success rate of instance I3-1 obtained by STHGA is 0.03. At the same time, the large number of function evaluations indicates that the searching efficiency reduces due to the curse of dimensionality.

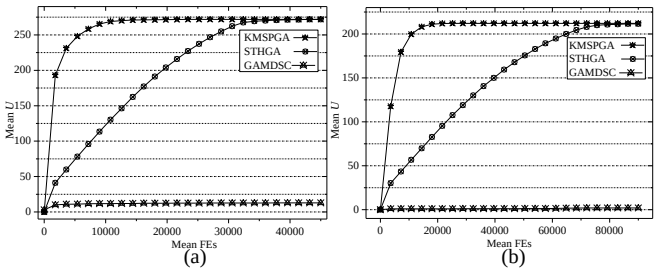


Fig. 13. Convergence curves of the compared algorithms. (a) I3-2. (b) I6-1.

Fig. 13 shows the convergence curves of the compared algorithms on instances I3-2 and I6-1, where the *x*-axis represents the mean FEs and the *y*-axis the mean *U* over 30 trials. The convergence rate of KMSPGA is higher than STHGA and GAMDSC. Furthermore, KMSPGA generates higher-quality solutions. In Fig. 13 (a), KMSPGA obtains the optimal value at about 15,000 FEs, STHGA reaches the near optimal value at 35,000 FEs, and GAMDSC evolves very slowly with a low-quality solution. Similarly, in Fig. 13 (b), KMSPGA converges to the optimal value at about 20,000 FEs, STHGA obtains the near optimal value at about 80,000 FEs, and GAMDSC still evolves very slowly with a low-quality solution.

In this subsection, we investigate the effectiveness and reliability of KMSPGA and the other compared algorithms with the same maximum number of function evaluations so that the comparison is fair. Some of the compared algorithms, e.g., STHGA, are not suitable for parallelism. The reasons are as follow. As can be noted from our description of the proposed parallel genetic framework, the disjoint cover sets within the same chromosome are supposed to be peer to each other. In STHGA, C_i is a complete cover set while C_{U+1} is incomplete due to the forward encoding scheme, which makes C_i ($i \in \{1, 2, \ldots, U\}$) is not peer to cover set C_{U+1}. Combination between any $C_{p,i}$ and $C_{q,j}$ ($i, j < U$) makes no sense to these already complete cover sets. The complicated auxiliary search

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2511142, IEEE Transactions on Evolutionary Computation

TABLE III
EXPERIMENTAL RESULTS ON 12 INSTANCES IN COMPARISON WITH PARALLEL ALGORITHMS OF A 2×4 PARTITION.

INSTANCE	PGA				KMPGA				SPGA				KMSPGA			
	FEs	U		S_r	FEs	U		S_r	FEs	U		S_r	FEs	U		S_r
	Mean	*Mean*	*Std*		*Mean*	*Mean*	*Std*		*Mean*	*Mean*	*Std*		*Mean*	*Mean*	*Std*	
I1-1	15000	28.7	1.34	0.00	14719	31.37	1.24	0.17	15000	28.9	1.73	0.00	**2688**	**33**	0	**1.00**
I1-2	15000	71.0	2.06	0.00	1500	78	1.68	0.00	4114	**83**	0	**1.00**	**2922**	**83**	0	**1.00**
I2-1	30000	63.5	2.19	0.00	29998	68.4	1.04	0.13	14688	69.8	0.50	0.80	**8418**	**70**	0	**1.00**
I2-2	30000	119.5	5.36	0.00	30000	159.8	3.77	0.00	29640	176.7	1.18	0.03	**9324**	**179**	0	**1.00**
I3-1	45000	83.2	3.85	0.00	45000	96.7	2.03	0.00	45000	100.3	2.12	0.00	**28350**	**105.8**	0.55	**0.87**
I3-2	45000	160.7	6.30	0.00	45000	231.3	6.01	0.00	43619	270.4	1.30	0.23	**15488**	**272**	0	**1.00**
I4-1	60000	106.1	4.90	0.00	60000	122.4	3.01	0.00	56098	144.2	1.19	0.17	**33086**	**145.9**	0.25	**0.93**
I4-2	60000	156.0	6.24	0.000	60000	290	3.06	0.00	60000	358.2	4.03	0.00	**35838**	**370**	0	**1.00**
I5-1	75000	137.8	3.96	0.00	75000	156.7	3.06	0.00	40688	175.8	0.41	0.80	**26761**	**175.7**	1.27	**0.93**
I5-2	75000	198.7	9.10	0.00	75000	370.1	6.10	0.00	74700	447.6	1.99	0.03	**36955**	**451**	0	**1.00**
I6-1	90000	138.8	6.18	0.00	90000	170.2	5.30	0.00	31485	211.9	0.18	0.97	**22845**	**212**	0	**1.00**
I6-2	90000	277.4	8.40	0.00	90000	428.3	7.66	0.00	39085	**528**	0	**1.00**	**34785**	**528**	0	**1.00**

TABLE IV
EXPERIMENTAL RESULTS ON 12 INSTANCES IN COMPARISON WITH PARALLEL ALGORITHMS OF A 4×4 PARTITION.

INSTANCE	PGA				KMPGA				SPGA				KMSPGA			
	FEs	U		S_r	FEs	U		S_r	FEs	U		S_r	FEs	U		S_r
	Mean	*Mean*	*Std*		*Mean*	*Mean*	*Std*		*Mean*	*Mean*	*Std*		*Mean*	*Mean*	*Std*	
I1-1	14900	28.9	1.76	0.03	14597	31.8	0.86	0.20	14920	29.4	1.57	0.03	**3660**	**33**	0	**1.00**
I1-2	15000	70.8	2.59	0.00	15000	78.7	1.57	0.00	6414	82.9	0.25	0.93	**4314**	**83**	0	**1.00**
I2-1	30000	65.3	1.54	0.00	29599	65.8	3.93	0.07	25997	68.5	1.07	0.20	**3960**	**70**	0	**1.00**
I2-2	30000	135.7	5.02	0.00	30000	170.0	2.03	0.00	30000	175.1	1.89	0.00	**11748**	**179**	0	**1.00**
I3-1	45000	90.4	2.33	0.00	45000	101.1	1.46	0.00	45000	99.6	1.85	0.00	**13230**	**106**	0	**1.00**
I3-2	45000	198.1	6.48	0.00	45000	258.7	3.42	0.00	44580	268.7	1.91	0.03	**23876**	**271.9**	0.18	**0.97**
I4-1	60000	129.2	2.85	0.00	60000	132.6	9.10	0.00	60000	141.6	2.04	0.00	**22782**	**145.9**	0.25	**0.93**
I4-2	60000	219.4	9.36	0.000	60000	340.0	4.69	0.00	60000	361	2.36	0.00	**29412**	**370**	0	**1.00**
I5-1	75000	169.2	2.36	0.00	72395	174.8	1.28	0.30	31588	175.8	0.41	0.80	**8625**	**176**	0	**1.00**
I5-2	75000	305.1	8.55	0.00	75000	410.3	5.78	0.00	71599	449.1	1.53	0.20	**29802**	**451**	0	**1.00**
I6-1	90000	203.2	2.57	0.00	89399	209.7	1.70	0.07	25586	211.9	0.18	0.97	**9975**	**212**	0	**1.00**
I6-2	90000	453.3	5.29	0.00	90000	504.5	4.73	0.00	26985	**528**	0	**1.00**	**24285**	**528**	0	**1.00**

operations adopted by STHGA also increases the difficulty of parallelizing the algorithm.

C. Comparison with Parallel Algorithms

In this subsection, we compare KMSPGA with the pure PGA, KMPGA, and SPGA. The PGA combined only with the KM combination or RTSS form KMPGA or SPGA. The experimental instances in Subsection *B* are adopted here. We conduct the experiments under two different partitions: $P_{2\times4}$ and $P_{4\times4}$.

The experimental results are given in Table III and Table IV. KMSPGA outperforms KMPGA and SPGA in all of the instances. Although S_r of KMPGA and SPGA is 0 in the majority of the instances, the mean U obtained by KMPGA and SPGA is much larger than PGA, which reveals that the KM combination and RTSS contribute to the enhancement of the solution quality, which is made available by KMSPGA. Take instance I4-2 of partition 2×4 as an example, where $\hat{U}$ is 370. The mean U obtained by PGA is 156.0, accounting for only 42.16%. As for KMPGA and SPGA, the mean U obtained are 290 and 358.2, accounting for 78.38% and 96.81%. SPGA obtains larger mean U than KMPGA in the majority of the instances, and therefore contribution of RTSS is larger than that of KM combination when it comes to the degree of the improvement of solution quality.

It is worth mentioning that parallel evolutionary algorithms are suitable for the problems of a high dimensionality or of complex and time-consuming computation features [51], such as large-scale air traffic flow optimization [52], discrete resource allocation in classic economic field [53], and large-scale function optimization [54][55]. They are adopted to either speed up the optimization or enhance the solution quality through a dimensionality reduction strategy. In this paper, the dimensionality and computational complexity of *Set K-cover* problem become so high in large-scale WSN that we adopt a parallel evolutionary algorithm for performance enhancement.

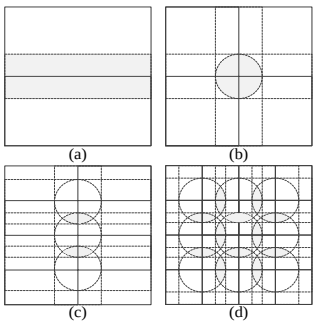


Fig. 14. Visual illustration of correlative areas under representative partitions. Grey areas represent the special correlative areas where sensors are relevant with the largest number of subareas. Sensors falling in grey areas are correlation with 2, 4, 6 and 6 subareas in (a), (b), (c), and (d) separately.

D. Discussion on Correlation between Sensors and Subareas

We investigate how the number of partitions affects the solution quality in this subsection. It can be intuitively noted that the number of subareas covered by one sensor increases as the target area is divided more finely. A larger number of subareas covered by one sensor means a higher correlation between the sensors and subareas. The high correlation influence the performance of parallel processing module and the KM combination. Fig. 14 gives a visual illustration of a special kind of correlative area (marked as grey), and sensors

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2511142, IEEE Transactions on Evolutionary Computation

TABLE V
EXPERIMENTAL RESULTS UNDER DIFFERENT PARTITIONS ON 12 INSTANCES

INSTANCE	$P_{2\times2}$ (ω_{-5}= 1.38, ω_{-8}=1.67)				$P_{2\times4}$ (ω_{-5}= 1.79, ω_{-8}=2.41)				$P_{4\times4}$ (ω_{-5}=2.30 , ω_{-8}=3.43)				$P_{4\times8}$ (ω_{-5}= 3.41, ω_{-8}=5.55)			
	FEs	U		S_r	FEs	U		S_r	FEs	U		S_r	FEs	U		S_r
	Mean	*Mean*	*Std*		*Mean*	*Mean*	*Std*		*Mean*	*Mean*	*Std*		*Mean*	*Mean*	*Std*	
I1-1	4965	**33**	0	**1.00**	**2688**	**33**	0	**1.00**	3660	**33**	0	**1.00**	4935	**33**	0	**1.00**
I1-2	2730	**83**	0	**1.00**	2922	**83**	0	**1.00**	4314	**83**	0	**1.00**	**2016**	**83**	0	**1.00**
I2-1	14604	69.9	0.57	0.83	8418	**70**	0	**1.00**	**3960**	**70**	0	**1.00**	4650	**70**	0	**1.00**
I2-2	9010	**179**	0	**1.00**	9324	**179**	0	**1.00**	11748	**179**	0	**1.00**	**3744**	**179**	0	**1.00**
I3-1	41490	104	1.24	0.37	28350	105.8	0.55	0.87	**13230**	**106**	0	**1.00**	15159	**106**	0	**1.00**
I3-2	3718	**272**	0	**1.00**	**15488**	**272**	0	**1.00**	23876	271.9	0.18	0.97	37464	265.1	4.55	0.23
I4-1	59445	143.8	1.44	0.1	33086	**145.9**	0.25	**0.93**	**22782**	**145.9**	0.25	**0.93**	45330	145.7	0.65	0.8
I4-2	52500	369.7	0.71	0.76	35838	**370**	0	**1.00**	**29412**	**370**	0	**1.00**	57060	346.6	6.89	0.07
I5-1	67695	174.5	1.98	0.43	26761	175.7	1.27	0.93	**8625**	**176**	0	**1.00**	10230	**176**	0	**1.00**
I5-2	66675	449.3	1.82	0.37	36955	**451**	0	**1.00**	**29802**	**451**	0	**1.00**	71010	433.2	5.71	0.07
I6-1	87660	209.6	2.22	0.2	22845	**212**	0	**1.00**	**9975**	**212**	0	**1.00**	12084	212	0	**1.00**
I6-2	78615	526.1	2.83	0.43	34785	**528**	0	**1.00**	**24285**	**528**	0	**1.00**	68115	521.4	5.06	0.30

falling in the grey area will be correlate with 2, 4, 6 and 6 subareas in (a), (b), (c), and (d), respectively. A sensor covers at least two subareas when it falls into the correlative areas. The correlative areas influence the independence of the evolution of each sub-population. The average number of subareas covered by one sensor is adopted to reveal the degree of correlation between sensors and subareas. A Monte Carlo method is utilized to estimate this value. One hundred thousand sensors (i.e., N = 100,000) are randomly deployed into the target area, then the number of subareas covered by each sensor is computed. The average number of subareas covered by one sensor is computed as ω:

$$\omega = \frac{1}{N}\sum_{i=1}^{N} N_{sa}(S_i) \tag{22}$$

where $N_{sa}(S_i)$ is the number of subareas covered by sensor S_i. A larger value of ω means a stronger correlation between sensors and subareas. The target area is expected to be divided into more subareas in order to achieve dimensionality reduction as the number of sensors increases. However, it is unreasonable to increase the number of subareas without constraints, because the efficiency of combination strategy decreases as the number of subareas increases.

Table V lists the results with different partitions: $P_{2\times2}$, $P_{2\times4}$, $P_{4\times4}$, and $P_{4\times8}$, where ω_{-r} represents the value of ω with sensing radius r. $P_{4\times4}$ achieves the best performance in the majority of test instances in terms of convergence rate, solution quality, and success rate. The success rate is low considering the performance of $P_{2\times2}$ and $P_{4\times8}$. However, the reason is completely different. As for $P_{2\times2}$, the partition quantity is not enough to reduce the dimensionality to an acceptable level. On the contrary, the partition quantity of $P_{4\times8}$ is so large that the correlation between sensors and subareas becomes too high to apply the divide-and-conquer strategy. As can be noted in instance Ix-1 and Ix-2, the success rate of the former is clearly higher than the later because of the smaller value of ω in Ix-1 than that in Ix-2. Consequently, the number of partitions is restricted by the value of ω. In order to help the divide-and-conquer strategy work efficiently, the partition quantity is limited to an appropriate range.

E. Experiments with Different Redundant Rate

In this subsection, we study the influence of different redundant rates on the solution quality. Two groups of experimental instances are adopted in this experiment, where the sensor radius is 5. Instances prefixed by "J" represent the number of sensors is 20,000, whereas instances prefixed by "H" represent the number of sensors is 25,000. Although the number and the sensing radius of sensors are fixed, the redundant rate can be different because of the random deployment strategy. It is quite difficult to generate an instance with a specified $\hat{U}$. Instead, to generate this test set with different redundant rates, we create a relatively large number of candidate instances, calculate their $\hat{U}$s, and then select the candidate instances with appropriate $\hat{U}$ to the test set. The redundant rate ranges from 3.997 to 5.003.

Performance of different partitions, i.e. $P_{2\times4}$, $P_{4\times4}$, and $P_{4\times8}$, is tested. STHGA is also adopted for comparison. Results are summarized in Table VI. KMSPGA ($P_{2\times4}$, $P_{4\times4}$, and $P_{4\times8}$) achieves a high success rate in a large range of redundant rates, which indicates that KMSPGA offers very promising performance with robustness. $P_{4\times4}$ achieves the fastest convergence rate, the largest mean U, and the highest S_r in the majority of the instances in Table VI. As far as STHGA is concerned, the success rate declines sharply when the redundant rate decreases. The x-axis of Fig. 15 (a) and (b) is the redundant rate, the y-axis of Fig.15 (a) represents mean S_r, and the y-axis of Fig. 15 (b) represents the ratio of mean U to $\hat{U}$. The best solution obtained by KMSPGA among different partitions in each instance is represented by P_{BEST}. In Fig. 15 (a), KMSPGA ($P_{2\times4}$, $P_{4\times4}$, and $P_{4\times8}$) obtains a high S_r in most of the instances while the S_r of STHGA declines sharply as η decreases. Fig. 15 (b) also indicates that KMSPGA possesses high solution quality within a large-scale range of η. In fact, P_{BEST} maintains a value of 1.00 in both Fig. 15 (a) and (b).

F. Parameter Investigation

We investigate the penalty coefficient λ and the threshold value of state factor ζ_c in this subsection in order to show how these parameters influence the performance of KMSPGA.

The penalty coefficient λ is adopted to improve the convergence rate. We adopt test instances I4-2 and I5-1 with partition $P_{4\times4}$. The value of λ is increased from 0 to 1 with step 0.05. Based on the experimental results, we give the relationship between λ and FEs in Fig. 16. As can be noted from the results, the value of FEs when λ is set within [0.05, 0.95], the required FEs is significantly less than that when λ is set to 1 (no penalty). The experimental results indicate that the penalty coefficient λ has an effect of improving the convergence rate of KMSPGA. Meanwhile, KMSPGA is generally insensitive to the value of λ, and it works identically well when λ is set to [0.2, 0.7].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2511142, IEEE Transactions on Evolutionary Computation

TABLE VI
EXPERIMENTAL RESULTS WITH DIFFERENT REDUNDANT RATES ON 20 INSTANCES

INSTANCE			STHGA				$P_{2\times4}$				$P_{4\times4}$				$P_{4\times8}$			
Na.	$\hat{U}$	η	FEs	U		S_r	FEs	U		S_r	FEs	U		S_r	FEs	U		S_r
			Mean	*Mean*	*Std*		*Mean*	*Mean*	*Std*		*Mean*	*Mean*	*Std*		*Mean*	*Mean*	*Std*	
J01	129	4.871	51407	128.8	0.38	0.83	2824	**129**	0	**1.00**	**2508**	**129**	0	1.00	3394	**129**	0	**1.00**
J02	133	4.724	52332	132.9	0.25	0.93	5802	**133**	0	**1.00**	**3684**	**133**	0	1.00	3904	**133**	0	**1.00**
J03	135	4.654	53898	134.8	0.38	0.83	6334	**135**	0	**1.00**	**4176**	**135**	0	1.00	6222	**135**	0	**1.00**
J04	136	4.620	56295	135.8	0.50	0.80	9714	**136**	0	**1.00**	**5856**	**136**	0	1.00	6664	**136**	0	**1.00**
J05	139	4.520	54113	138.8	0.38	0.83	9210	**139**	0	**1.00**	**7410**	**139**	0	1.00	7485	**139**	0	**1.00**
J06	145	4.333	59789	143.1	1.05	0.06	18705	**145**	0	**1.00**	**13350**	**145**	0	1.00	15204	**145**	0	**1.00**
J07	149	4.217	59281	147.8	1.07	0.30	29544	**149**	0	**1.00**	16740	**149**	0	1.00	**10512**	**149**	0	**1.00**
J08	155	4.080	60000	146.1	1.58	0.00	60000	150.5	2.13	0.00	**25875**	**155**	0	1.00	53992	154.0	1.48	0.53
J09	157	4.002	60000	152	1.69	0.00	57819	155.5	1.57	0.37	**35235**	**157**	0	1.00	59925	151.7	2.63	0.03
J10	158	3.977	60000	154.0	1.52	0.00	37884	**158**	0	**1.00**	**33045**	157.9	0.40	0.93	57420	156.2	1.21	0.17
H01	157	5.003	56796	**157**	0	**1.00**	**2674**	**157**	0	**1.00**	4680	**157**	0	1.00	4035	**157**	0	**1.00**
H02	159	4.940	62349	158.9	0.35	0.86	**3524**	**159**	0	**1.00**	4485	**159**	0	1.00	5250	**159**	0	**1.00**
H03	162	4.848	63270	161.9	0.31	0.90	**3954**	**162**	0	**1.00**	4485	**162**	0	1.00	4635	**162**	0	**1.00**
H04	164	4.789	63223	163.9	0.31	0.90	**5054**	**164**	0	**1.00**	5085	**164**	0	1.00	5805	**164**	0	**1.00**
H05	167	4.703	67287	166.9	0.35	0.87	7305	**167**	0	**1.00**	**4635**	**167**	0	1.00	6030	**167**	0	**1.00**
H06	173	4.540	72310	172.5	0.63	0.57	15330	**173**	0	**1.00**	**7632**	**173**	0	1.00	16059	**173**	0	**1.00**
H07	175	4.488	73608	174.1	0.92	0.40	19980	**175**	0	**1.00**	**11430**	**175**	0	1.00	19960	**175**	0	**1.00**
H08	178	4.412	74741	176.4	1.04	0.10	19685	**178**	0	**1.00**	**10932**	**178**	0	1.00	15810	**178**	0	**1.00**
H09	181	4.340	74003	179.9	0.96	0.33	20730	**181**	0	**1.00**	**10575**	**181**	0	1.00	15510	**181**	0	**1.00**
H10	185	4.245	74570	183.4	1.10	0.13	41559	**185**	0	**1.00**	**18375**	**185**	0	1.00	24435	**185**	0	**1.00**

The threshold value of the state factor, ζ_c, determines the frequency of executing the KM operation. We adopt test instance I2-2 to investigate the effect of this parameter. For partition $P_{2\times4}$ and $P_{4\times4}$, ζ_c is increased from 0 to 7 with step 0.5 and from 0 to 15 with step 1, respectively. The experimental results show that, KMSPGA achieves a 100% success rate with different ζ_c. Therefore, ζ_c has no influence on the solution quality. Fig. 17 shows the influence of the value of ζ_c on the FEs and a practical execution time of KMSPGA. It can be observed that ζ_c has negligible influence on the convergence rate. Based on the experimental results, we suggest that the value of ζ_c is set to $\lfloor 0.94 \times L \times W \rfloor$.

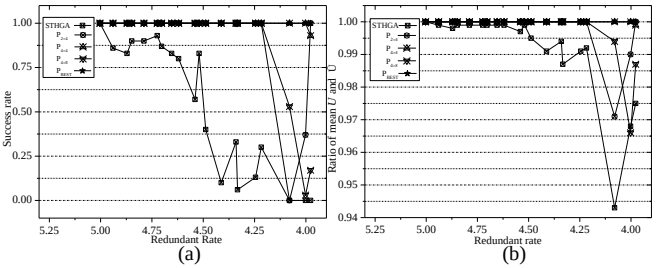


Fig. 15. Influence of redundant rate on the performance of KMSPGA. (a) Influence on the mean success rates. (b) Influence on the mean ratios.

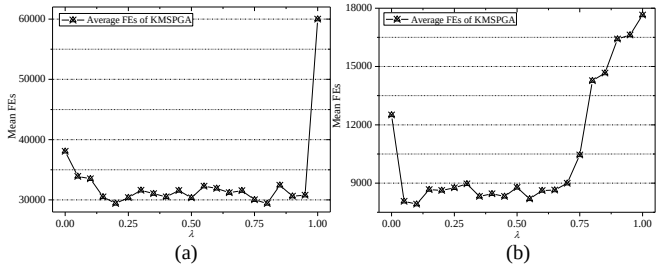


Fig. 16. Influence of λ on the performance of KMSPGA. (a) I4-2. (b) I5-1.

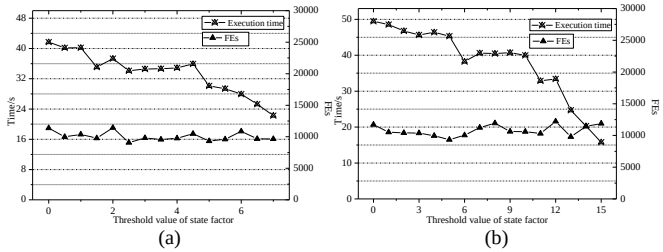


Fig. 17. Influence of threshold value ζ_c on the performance of KMSPGA. (a) $P_{2\times4}$. (b) $P_{4\times4}$.

G. Proof-of-Principle Experiments

To conduct proof-of-principle experiments, we first apply a deterministic deployment strategy to generate K complete cover sets. The superposition of these K complete cover sets results in a *Set K-cover* instance, to which an optimal solution is known. Then KMSPGA and other compared algorithms are applied to solving this instance.

Here, it is to be noted that deterministically deploying the least number of circles to cover any polygon is an NP-hard problem as discussed in [21], [56], and [57]. Nevertheless, if we relax the requirement of that the layout should be "theoretically best with the least circles", there exists an efficient deterministic node placement strategy to cope with this issue. Fig. 18 shows a "tessellation" placement strategy. First, the target area is completely tiled by a number of compact polygons. Then, sensors are placed at the vertices of polygons. If each polygon is covered by at least one sensor, the target area is completely covered. Fig. 18 is adopted as 1-cover set (unit set) since it is the optimal tessellation requiring the minimum number of sensors. We then repeatedly add K unit sets into a cover set to generate a K-cover as the instance of the proof-of-principle experiment. Six new instances are

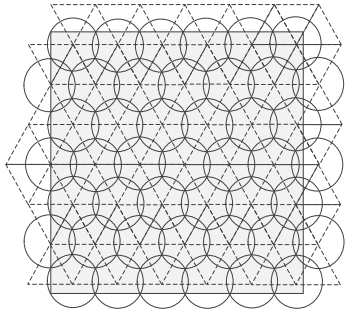


Fig. 18. Illustration of "tessellation" placement strategy, and the polygons adopted here is triangle.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2511142, IEEE Transactions on Evolutionary Computation

TABLE VII
EXPERIMENTAL RESULTS OF THE PROOF-OF-PRINCIPLE EXPERIMENT ON 6 INSTANCES

INSTANCE	R	Û	GAMDSC					STHGA					KMSPGA			
			FEs		*U*	S_r	*t-test*[1]	FEs		*U*	S_r	*t-test*[2]	FEs		*U*	S_r
			Mean	*Std*	*Mean*		*p-value*	*Mean*	*Std*	*Mean*		*p-value*	*Mean*	*Std*	*Mean*	
DD1-1	5	100	13500	0	1	0	4.14e-80	598	0.46	**100**	**1.00**	1.09e-22	**463.5**	25.94	**100**	**1.00**
DD1-2	8		6600	0	4.1	0	3.31e-70	598	0.58	**100**	**1.00**	1.09e-30	**328.5**	27.39	**100**	**1.00**
DD2-1	5	300	40500	0	1	0	1.30e-86	1798	0.72	**300**	**1.00**	3.13e-40	**779.5**	47.13	**300**	**1.00**
DD2-2	8		19800	0	6.2	0	3.07e-78	1798	0.50	**300**	**1.00**	2.56e-44	**494.0**	44.56	**300**	**1.00**
DD3-1	5	600	81000	0	1	0	5.18e-82	3598	0.52	**600**	**1.00**	1.87e-37	**1285.5**	136.35	**600**	**1.00**
DD3-2	8		39600	0	8.0	0	8.54e-82	3598	0.46	**600**	**1.00**	3.08e-49	**661.0**	67.76	**600**	**1.00**

TABLE VIII
EXPERIMENTAL RESULTS WITH SENSORS OF DIFFERENT RADIUSES ON 5 INSTANCES

INSTANCE	Û	GAMDSC					STHGA					KMSPGA			
		FEs	*U*		*t-test*[1]	S_r	FEs	*U*		*t-test*[2]	S_r	FEs	*U*		S_r
		Mean	*Mean*	*Std*	*p-value*		*Mean*	*Mean*	*Std*	*p-value*		*Mean*	*Mean*	*Std*	
DR1	159	45000	5.1	0.48	1.98e-74	0	44815	157.7	0.71	3.6e-11	0.1	**11714**	**159**	0	**1.00**
DR2	220	45000	8.1	0.71	1.65e-73	0	44148	219.0	0.96	2.28e-06	0.33	**12954**	**220**	0	**1.00**
DR3	276	45000	15.8	0.92	8.4e-73	0	41145	275.6	0.61	2.8e-03	0.7	**14794**	**276**	0	**1.00**
DR4	163	45000	5.4	0.50	4.0e-74	0	40232	162.8	0.40	1.2e-02	0.8	**6914**	**163**	0	**1.00**
DR5	232	45000	7.7	0.58	4.31e-75	0	44762	230.7	0.99	8.3e-07	0.16	**25714**	**231.9**	0.25	**0.93**

1 Two tailed t-test of the null hypothesis that GAMDSC is equal to KMSPGA. 2 Two tailed t-test of the null hypothesis that STHGA is equal to KMSPGA.

TABLE IX
EXPERIMENTAL RESULTS IN 3-D ENVIRONMENT ON 6 INSTANCES

INSTANCE	N	R_s	Û	GAMDSC					STHGA					KMSPGA			
				FEs	*U*		*t-test*[1]	S_r	FEs	*U*		*t-test*[2]	S_r	FEs	*U*		S_r
				Mean	*Mean*	*Std*	*p-value*		*Mean*	*Mean*	*Std*	*p-value*		*Mean*	*Mean*	*Std*	
3D1-1	5000	5	25	15000	1	0	0	0	12673	24.9	0.31	8e-02	0.9	**4785**	**25**	0	**1.00**
3D1-2		8	76	15000	3.5	0.51	2.84e-64	0	13449	75.9	0.25	16e-01	0.93	**4618**	**76**	0	**1.00**
3D2-1	10000	5	54	30000	1	0	4.91e-69	0	30000	50.5	0.97	8.3e-18	0	**16309**	**53.9**	0.25	**0.93**
3D2-2		8	163	30000	3.2	0.43	4.12e-67	0	30000	157.2	1.31	2.09e-18	0	**20756**	**162.5**	0.62	**0.67**
3D3-1	15000	5	84	45000	1	0	2.34e-63	0	45000	75.1	1.18	1.70e-25	0	**41736**	**83.6**	0.62	**0.67**
3D3-2		8	250	45000	3.5	0.51	1.35e-66	0	45000	234.7	2.17	5.03e-24	0	**34290.5**	**249.1**	1.35	**0.63**

1 Two tailed t-test of the null hypothesis that GAMDSC is equal to KMSPGA. 2 Two tailed t-test of the null hypothesis that STHGA is equal to KMSPGA.

generated in this way with K setting to 100, 300, and 600. The numbers of sensors covering the target area are 45 and 22 when R_s is 5 and 8, respectively.

The experimental results are given in Table VII. Both STHGA and KMSPGA achieve 100% success rate in the tests. However, KMSPGA achieves a higher convergence rate than STHGA. The experimental results also indicate that these ideal and regular test instances are easy to solve by the two algorithms. The reasons are presented as follow. Aiming at using the least number of circles to realize complete coverage, the deterministic deployment strategy tends to minimize the overlapping area of neighboring circles. Thus, for each sensor, the coverage ratio of each cover set will be quite different considering whether or not the sensor is assigned to the right cover set. This feature makes the fitness evolution possess good differentiation for different individuals (candidate solutions) and hence provides a promising guidance for the search. Furthermore, this feature also benefits the proposed redundancy-based schedule strategy, owing to the uniqueness of sensor within each unit set.

H. Experiments with Sensors of Different Radiuses

Although in the above experiments, sensors of identical radius are assumed for simplicity. However, as the proposed KMSPGA algorithm does not contain any radius-related parameters or operators, it is a generic algorithm suitable for both application scenarios with homogenous or heterogeneous sensors deployed. In this subsection, we conduct experiments using sensors of different radiuses to investigate the performance of KMSPGA. Five new instances are generated and tested, in which the radiuses of sensors follow Gaussian distribution with different mean values and standard deviations. The radius of sensor j in instance i is set to R_i+r_j, where r_j is a random number following a standard normal distribution. For $i \in \{0, 1, 2\}$, R_i is set to 6, 7, and 8, respectively. For $i = 3$ and $i = 4$, R_i is randomly chosen within an integer interval [5, 7] and [6, 8], respectively. The number of each instance is 15000.

From the experimental results given in Table VIII, it is still observed that KMSPGA achieves significantly higher solution quality and success rate among the compared algorithms in most of the instances. Besides, KMSPGA achieves smallest standard deviations, which indicates that KMSPGA possesses high stability. The results confirm that KMSPGA works well with sensors of different radiuses.

I. Experiment in a 3-D Environment

In the literature of WSN lifetime maximization, sensors are assumed to be uniformly distributed on an ideal square plane. However, in practice, this is not always the case. Instead, sensors are often deployed on a 3-D surface so that the sensor distribution is no longer uniform, but it is highly dependent to the shape/landscape of the surface [58]. This uneven distribution makes it even harder to optimally schedule the sensors into the right complete cover sets. In this subsection, we implement KMSPGA to test its effectiveness and reliability on this application scene.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2015.2511142, IEEE Transactions on Evolutionary Computation

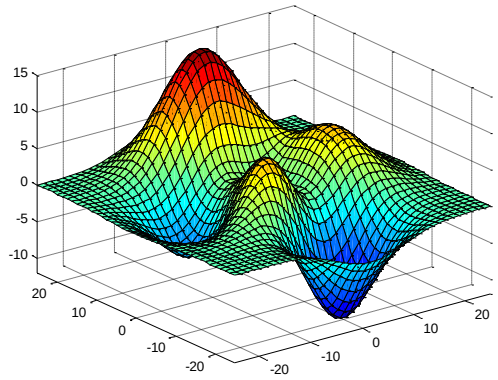


Fig. 19. Simulated free flowing contour.

The multi-peak function used to simulate the 3-D surface is given as

$$\Gamma_{3-D}(x,y) = (3\cdot(1-\frac{x}{10})^2\cdot e^{-(\frac{x}{10})^2-(\frac{y}{10}+1)^2} - 10\cdot(\frac{x}{50} - (\frac{x}{10})^3 - (\frac{y}{5})^5)\cdot e^{-(\frac{x}{10})^2-(\frac{y}{10})^2} - \frac{1}{3}\cdot e^{-(\frac{x}{10}+1)^2-(\frac{y}{10})^2})\cdot 2 \quad (22)$$

which consists of three peaks and three valleys. Its 3-D view is shown in Fig. 19. The number of sensors is set to 500, 10000, and 15000. GAMDSC and STHGA are also modified to solve the *Set K-Cover* problem in this 3-D environment. Experimental results given in Table IX indicate that these instances are significantly hard to solve. The performance of all the algorithms decreases on this test set. GAMDSC and STHGA are unable to achieve an acceptable solution within the limited FEs. In comparison, the proposed KMSPGA still offers a high success rate and solution quality, which further confirms the effectiveness and reliability of KMSPGA.

V. Conclusion

Due to the curse of dimensionality, existing set cover algorithms are unable to provide satisfactory efficiency for large-scale WSNs scheduling. In this paper, we have developed a Kuhn-Munkres parallel genetic algorithm, KMSPGA, for the set cover problem and applied it to lifetime maximization of large-scale WSNs. The KMSPGA framework is based on a divide-and-conquer strategy of dimensionality reduction. Firstly, the target area is divided into several subareas, and then individuals are evolved independently in each subarea until the state factor reaches a predefined value. The polynomial Kuhn-Munkres algorithm is then utilized to splice the solutions obtained in each subarea so as to generate global optimal solution of the entire problem. KMSPGA also includes a novel schedule operation for further improvement in performance.

Eight types of experiments have been conducted to verify the design and effectiveness of KMSPGA. The experimental results indicate that KMSPGA achieves a higher convergence rate, solution quality, success rate, and scalability. Further, by investigating the influence of different partitions, redundant rate, penalty coefficient on the performance of KMSPGA, KMSPGA is also seen to offer high robustness. Finally, experimental results on new testing scenarios indicate that KMSPGA achieves wide applicability.

Future work includes the development of an improved matching algorithm for the combination operation, of distributed, cloud and multi-objective versions of KMSPGA and their applications to various kinds of real-world problems.

References

[1] J. Guevara, F. Barrero, E. Vargas, J. Becerra, and S. Toral, "Environmental wireless sensor network for road traffic applications," *IET Intelligent Transport Systems*, vol. 6, no. 2, pp. 177-186, Jun. 2012.

[2] R. Mittal and M. P. S. Bhatia, "Wireless Sensor Networks for monitoring the environmental activities," in *Proceedings of IEEE International Conference on Computational Intelligence and Computing Research*, pp. 1-5, 2010.

[3] J. G. Ko, C. Lu, M. B. Srivastava, J. A. Stankovic, A. Terzis, and M. Welsh, "Wireless sensor networks for healthcare," in *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1947-1960, 2010.

[4] M. C. Rodríguez-Sánchez, S. Borromeo, and J. A. Hernández-Tamames, "Wireless sensor networks for conservation and monitoring cultural assets," *IEEE Sensors Journal*, vol. 11, no. 6, pp. 1382-1389, Jun. 2011.

[5] K. Römer, F. Mattern, and E. Zurich, "The design space of wireless sensor networks," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 54-61, 2004.

[6] G. Anastasi, M. Conti, and M. D. Francesco, "Extending the lifetime of wireless sensor networks through adaptive sleep," *IEEE Transactions on Industrial Informatics*, vol. 5, no. 3, pp. 351-365, 2009.

[7] V. Shah-Nansouri and V. W. S. Wong, "Lifetime-resource tradeoff for multicast traffic in wireless sensor networks," *IEEE Transaction on Wireless Communication*, vol. 9, no. 6, pp. 1924-1934, 2010.

[8] X. -Y. Tang and J. -L. Xu, "Optimizing lifetime for continuous data aggregation with precision guarantees in wireless sensor networks," *IEEE Transactions on Networking*, vol. 16, no. 4, pp. 904-917, 2008.

[9] C. -F. Wang, I. -D. Shih, B. -H. Pan, and T. -Y. Wu, "A network lifetime enhancement method for sink relocation and its analysis in wireless sensor networks," *IEEE Sensors Journal*, vol. 14, no. 6, pp. 1932-1943, 2014.

[10] M. N. Rahman and M. A. Matin, "Efficient algorithm for prolonging network lifetime of wireless sensor networks," *Tsinghua Science and Technology*, vol. 16, no. 6, pp. 561-568, 2011.

[11] W. Wang, V. Srinivasan, and K. -C. Chua, "Extending the lifetime of wireless sensor networks through mobile relays," *IEEE/ACM Transactions on Networking*, vol. 16, no. 5, 2008.

[12] K. Pradeepa, W. R. Anne, and S. Duraisamy, "Improved sensor network lifetime using multiple mobile sinks: a new predetermined trajectory," in *Proceedings of International Conference on Computing Communication and Networking Technologies*, pp. 1-6, 2010.

[13] A. A. Aziz, Y. A. Sekercigolu, P. Fitzpatrick, and M. Ivanovich, "A survey on distributed topology control techniques for extending the lifetime of battery powered wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 121-144, 2013.

[14] D. P. Dahnil, Y. P. Singh, and C. K. Ho, "Topology-controlled adaptive clustering for uniformity and increased lifetime in wireless sensor networks," *Wireless Sensor Systems*, vol. 2, no. 4, pp. 318-327, 2012.

[15] I. S. AlShawi, L. Yan, W. Pan, and B. Luo, "Lifetime enhancement in wireless sensor networks using fuzzy approach and a-star algorithm," *IEEE Sensors Journal*, vol. 12. No. 10, pp. 3010-3018, 2012.

[16] J. –H. Chang and L. Tassiu, "Maximizing lifetime routing in wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 4, pp. 609-619, 2004.

[17] I. F. Akyildiz, W. -L. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102-114, 2002.

[18] J. -M. Chen, E. -T. Shen, and Y. -X. Sun, "The deployment algorithms in wireless sensor networks: a survey," *Information Technology Journal*, pp. 293-301, 2009.

[19] G. -J. Fan and S. -Y. Jin, "Coverage problem in wireless sensor networks: a survey," *Journal of Networks*, vol. 5, no. 9, pp. 1033-1040, Sept. 2010.

[20] H. -T Zhang and C. Liu, "A review on node deployment of wireless sensor networks," *International Journal of Computer Science Issues*, vol. 9, no. 3, Nov. 2013.

[21] Y. Yoon and Y. -H. Kim, "An efficient genetic algorithm for maximum coverage deployment in wireless sensor networks," *IEEE Transactions on Cybernetics*, vol. 43, no. 5, pp. 1473-1483, Dec. 2013.

[22] S. -B. He, X. -W. Gong, J. -S. Zhang, J. -M. Chen, and Y. -X. Sun, "Curve-based deployment for barrier coverage in wireless sensor networks," *IEEE Transactions on Communications*, vol. 13, no. 2, pp. 724-735, Feb. 2014.

[23] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization in distributed sensor networks," *ACM Transactions on Embedded Computing Systems*, vol. 3, no. 1, pp. 61-91, 2004.

[24] R. Kershner, "The number of circles covering a set", *American Journal of Mathematics*, vol. 61, no. 3, pp. 665-671, Jul. 1939.

[25] S. Verblunsky, "On the least number of unit circles which can cover a square", *Journal of London Mathematical Society*, vol. 24, no. 3, pp. 164-170, 1949.

[26] T. Tabirca, L. T. Yang, and S. Tabira, "Smallest number of sensors for *k*-covering", *International Journal of Computers Communications & Control*, vol. 8, no. 2, pp. 312-319, Apr. 2013.

[27] M. Ryerkerk, R. Averill, K. Deb, and E. Goodman, "Meaningful representation and recombination of variable length genomes", *Proceedings of Genetic and Evolutionary Computation Conference*, pp. 1471-1472, 2012.

[28] A. Makhoul and C. Pham, "Dynamic scheduling of cover-sets in randomly deployed wireless video sensor networks for surveillance applications," *Wireless Days*, pp. 1-6, Dec. 2009.

[29] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," *IEEE International Conference on Communications*, vol. 2, pp. 472-476, 2001.

[30] M. Cardei and D. -Z. Du, "Improving wireless sensor network lifetime through power aware organization," *Wireless Networks*, vol. 11, no. 3, pp. 333-340, May. 2005.

[31] Q. Wang, W. -J. Yan, and Y. Shen, "N-person card game approach for solving set k-cover problem in wireless sensor networks," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 5, pp. 1522-1535, May. 2012.

[32] O. E. David, H. J. van den Herik, M. Koppel, and N. S. Netanyahu, "Genetic algorithms for evolving computer chess programs," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 5, pp. 779-789, Oct. 2014.

[33] W. B. Langdon and M. Harman, "Optimizing existing software with genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 1, pp. 118-135, Feb. 2015.

[34] K. Seo, S. Hyun, and Y. -H. Kim, "An edge-set representation based on a spanning tree for searching cut space," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 4, Aug. 2015.

[35] D. Perez, J. Togelius, S. Samothrakis, and P. Rohlfshagen, "Automated map generation for the physical traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 5, pp. 708-720, Oct. 2014.

[36] G. M. Khan, R. Arshad, S. A. Mahmud, and F. Ullah, "Intelligent bandwidth estimation for variable bit rate traffic", *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 1, pp. 151-155, Fed. 2015.

[37] M. K. Marichelvam, T. Prabaharan, and X. -S. Yang, "A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 301-305, Apr. 2014.

[38] C. C. Lai, T. C. Kang, and K. R. Song, "An effective genetic algorithm to improve wireless sensor network lifetime for large-scale surveillance," *IEEE Congress on Evolutionary Computation*, pp. 3531-3538, 2007.

[39] X.-M. Hu, J. Zhang, and Y. Yu, "Hybrid genetic algorithm using a forward encoding scheme for lifetime maximization of wireless sensor networks," *IEEE Transactions on Evolutionary Computation*, pp. 766-781, 2010.

[40] Y. Lin, J. Zhang, H. S. -H. Chung, Y. Li, and Y. H. Shi, "An ant colony optimization approach for maximizing the lifetime of heterogeneous wireless sensor networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 42, no. 3, pp. 408-420, May 2012.

[41] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83-97, Mar. 1955.

[42] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32-38, 1957.

[43] B. Wang, "Coverage problems in sensor networks: a survey," *ACM Computing Surveys*, vol. 43, no. 4, pp. 32:1-32:53, Oct. 2011.

[44] A. Hossain, P. K. Biswas, and S. Chakrabarti, "Sensing models and its impact on network coverage in wireless sensor network," in *Proceedings of IEEE Region 10 and the Third international Conference on Industrial and Information Systems*, pp. 1-5, 2008.

[45] N. -N. Qin, F. Xu, J. Yang, and G. -S. Liang, "Research on the sensing model in wireless sensor networks," in *Proceedings of International Conference on Intelligent Computation Technology and Automation*, pp. 169-172, 2010.

[46] S. Pudasaini, S. Moh, and S. Seokjoo, "Stochastic coverage analysis of wireless sensor network with hybrid sensing model," in *Proceedings of International Conference on Advanced Communication Technology*, pp. 549-553, 2009.

[47] X. -R. Wang, G. -L. Xing, Y. -F. Zhang, C. -Y. L, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *Proceedings of the 1st International Conference on Embedded Sensor Systems*, pp. 28-39, 2003.

[48] K. Zheng, F. Liu, Q. Zheng, and W. Xiang, "A graph-based cooperative scheduling scheme for vehicular networks", *IEEE Transactions on Vehicular Technology*, vol. 62, no. 4, pp. 1450-1458, Feb. 2013.

[49] H. -B. Zhu, M. -C. Zhou, and R. Alking, "Group role assignment via a Kuhn-Munkres algorithm-based solution", *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 42, no. 3, pp. 739-750, Nov. 2011.

[50] X. -T. Zhou, L. -Q. Yang, and D. -F. Yuan, "Bipartite matching based user grouping for grouped OFDM-IDMA", *IEEE Transactions on Wireless Communications*, vol. 12, no. 10, pp. 5248-5257, Sep. 2013.

[51] Y. -J. Gong, W. -N. Chen, Z. -H. Zhan, et al., "Distributed evolutionary algorithms and their models: a survey of the state-of-the-art," *Applied Soft Computing*, vol. 34, pp. 286-300, Sep. 2015.

[52] Y. Cao and D. -F. Sun, "A parallel computing framework for large-scale air traffic flow optimization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1855-1864, Dec. 2012.

[53] B. B. M and H. R. Rao, "A parallel hypercube algorithm for discrete resource allocation problems," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 36, no. 1, Jan. 2006.

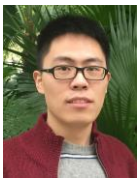
[54] X. -D. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 2, Apr. 2012.

[55] M. N. Omidvar, X. -D. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 378-393, Jun. 2014.

[56] M. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, San Francisco, CA, USA: Freeman, 1979.

[57] T. Tabirca, L. T. Yang, and S. Tabira, "Smallest number of sensors for *k*-covering", *International Journal of Computers Communications & Control*, vol. 8, no. 2, pp. 312-319, Apr. 2013.

[58] L. -H. Kong, M. -C. Zhao, X. -Y. Liu, and J. L. Lu, "Surface coverage in sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 234-243, Jan. 2014.



Xin-Yuan Zhang (S'14) received the B. S. degree from Sun Yat-sen University, China, in 2014, where he is currently pursuing the Ph. D. degree. His current research interests include evolutionary computation algorithms, swarm intelligence algorithms, their applications in real-world problems, and smart grid.

Yue-Jiao Gong (S'10-M'15) received the Ph.D. degree in Computer Science from Sun Yat-sen University, China, in 2014. She is currently a Post-Doctoral Research Fellow with the Department of Computer and Information Science, University of Macau, Macau.

Her research interests include evolutionary computation, swarm intelligence, and their applications to intelligent transportation scheduling, wireless sensor network, and image processing. She has published over 30 papers, including ten IEEE Trans. papers, in her research area. Dr.

Gong currently serves as a reviewer for IEEE Trans. on Evolutionary Computation, IEEE Trans. on Cybernetics, and IEEE Trans. on Intelligent Transportation Systems.

Zhi-Hui Zhan (S'09-M'13) received the Bachelor's degree and the Ph. D degree in 2007 and 2013, respectively, from the Department of Computer Science of Sun Yat-Sen University, Guangzhou, China. He is currently an associate professor with the School of Advanced Computing, Sun Yat-sen University.

His current research interests include evolutionary computation algorithms, swarm intelligence algorithms, and their applications in real-world problems, and in environments of cloud computing and big data. Dr. Zhan's doctoral dissertation was awarded the China Computer Federation Outstanding Dissertation in 2013. Dr. Zhan received the Natural Science Foundation for Distinguished Young Scientists of Guangdong Province, China in 2014 and was awarded the Pearl River New Star in Science and Technology in 2015. Dr. Zhan is listed as one of the Most Cited Chinese Researchers in Computer Science.

Wei-Neng Chen (S'07-M'12) received the Bachelor's degree and the Ph.D. degree from the Department of Computer Science of Sun Yat-sen University, Guangzhou, China, in 2006 and 2012, respectively. He is currently an associate professor with the School of Advanced Computing, Sun Yat-sen University, China.

His current research interests include swarm intelligence algorithms and their applications on cloud computing, financial optimization, operations research and software engineering. He has published more than 30 papers in international journals and conferences. His doctoral dissertation was awarded the China Computer Federation (CCF) outstanding dissertation in 2012.

Yun Li (S'87-M'90) received his Ph.D. in computing and control in 1990. He is currently a professor with Department of Systems Engineering, University of Glasgow, U.K. He served as Founding Director of University of Glasgow Singapore during 2011-2013 and acted as Founding Director of the University's international joint program with University of Electronic Science and Technology of China (UESTC) in 2013. He was invited to Kumamoto University, Japan, as Visiting Professor in 2002 and is currently Visiting Professor to UESTC and Sun Yat-sen University, China, researching into smart design with market informatics via the cloud to complete the value chain for Industry 4.0.

Dr. Li is an Associate Editor of the IEEE Trans. on Evolutionary Computation and of the SM Journal of Engineering Sciences. He has 200 publications and is a Chartered Engineer.

Jun Zhang (M'02–SM'08) received the Ph.D. degree from the City University of Hong Kong, Hong Kong, in 2002. He is currently a Changjiang Chair Professor with Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, China.

His research interests include computational intelligence, cloud computing, data mining, and power electronic circuits. He has published over 200 technical papers in his research area. Dr. Zhang was a recipient of the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011 and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE Trans. on Evolutionary Computation, IEEE Trans. on Industrial Electronics and IEEE Trans. on Cybernetics.